НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Геометричне моделювання в інформаційних системах»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Розробка інтерфейсу twain для веб-застосувань Cache»

Виконала:
студентка IV курсу, групи ТР-61
Кортельова Яна Олексіївна _____

Керівник:
к.т.н, доцент
Михайлова Ірина Юріївна _____

Рецензент:
доц., к.т.н., доц.
Побіровський Ю. М.

_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

Кортельова Яна Олексіївна
(прізвище, ім’я, по батькові)

1. Тема роботи «Twain інтерфейс для веб застосувань Cache»_____
_____
_____
керівник роботи_Михайлова Ірина Юріївна_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _____19 червня 2020 року_____
3. Вихідні дані до роботи ____Інтерфейс з доступом до бази даних Cache, веб-застосунок створений за допомогою мови JavaScript, Html, Css, Node.js Bootstrap, Express Framework_____
_____
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)_Виконати аналіз існуючих алгоритмів та систем розв’язання задачі, розробити базу даних проекту, опис розробленого інтерфейсу та веб-затсосунку, опис архітектури та алгоритму роботи розробеного програмного продукту
5. Перелік ілюстративного матеріалу Блок-схеми розроблених алгоритмів, схема архітектури розробленої програмної системи, діаграми прецедентів, ілюстарації роботи програми

6. Дата видачі завдання ”10” __листопада__2019_ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	10.12.19	
2.	Вивчення та аналіз задачі	24.01.20	
3.	Розробка архітектури та загальної структури системи	14.02.20	
4.	Розробка структур окремих підсистем	24.02.20	
5.	Програмна реалізація системи	03.03.20	
6.	Оформлення пояснювальної записки	06.06.20	
7.	Захист програмного продукту	07.06.20	
8.	Передзахист	08.06.20	
9.	Захист	16.06.20	

Студент _____ _____Кортельова Я.О._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Михайлова І.Ю._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою цієї роботи є створення TWAIN інтерфейсу для веб-застосувань Caché та сервісу фотохостингу з можливістю сканування зображень для демонстрації його роботи.

Загальна постановка задачі полягає в реалізації можливості доступу до сканера з вікна браузеру та отримання інформацію одразу ж в браузер, а також збереження інформації до бази даних InterSystems Caché.

Загальний обсяг роботи: 66 сторінок, 19 ілюстрацій та 18 бібліографічних найменувань.

Ключові слова: rest api, cache, intersystems, twain.

ABSTRACT

The objective of this work is to create a TWAIN interface for Caché web applications and photohosting service with possibility to scan images.

The general version of the task must be available to the scanner from a windowed browser and using browser access data as well as InterSystems Caché database information.

General processing of the work: 66 pages, 19 figures and 18 bibliographic titles.

Key words: rest api, cache, intersystems, twain.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

API – Application Interface Programming, програмний інтерфейс застосування.

URL – Uniform Resource Locator, уніфікований вказівник ресурсу.

REST- Representational State Transfer, передача стану представлення.

CRUD – Create Read Update Delete, операції створення, зчитування, зміни, видалення.

ODBC – Open Database Connectivity, API для доступу до БД.

DSN – Destination Source Name, ім’я джерела даних.

DOM – Document Object Model, програмний інтерфейс для доступу, який дозволяє програмам та скриптам отримувати доступ до HTML-, XHTML- та XML-документів, а також змінювати вміст, структуру та оформлення таких документів.

СУБД – система управління базами даних.

БД – база даних.

Twain – програмний інтерфейс, який визначає взаємодію між програмами та пристроями захвату зображення, таким як сканер або камера.

ВСТУП

Наразі майже кожен житель планети під'єднаний до інтернету. Всесвітнє павутиння являється найрозвиненішею технологією інтернет. Інтернет надає доступ до будь-якої інформації, можна сказати, що це велика бібліотека знань, яка поповнюється з кожним днем. Інформацію можна знайти використовуючи різноманітні пошукачі, відтак зрозуміло чому вони являються найпопулярнішими веб-сайтами в світі. Друге місце за популярністю посідають соціальні мережі.

Певна частина людського життя перемістилась саме в соціальні мережі. Люди мають можливість бути на зв'язку майже в кожній точці світу. В соціальних мережах вони спілкуються, знаходять друзів, роботу, поширюють інформацію про себе. Це дозволяє досить просто знайти собі нових знайомих з такими ж інтересами. Тобто така популярність соціальних мереж цілком логічна.

Однією із найпопулярніших соціальних мереж в Україні є Instagram. Основним елементом в ній є зображення. В середньому 'вага' зображення, враховуючи його розмір та розширення, може становити від 200 кб до 5 мб. Таким чином, зберігаючи фото на своєму пристрої, значно завантажується пам'ять жорстокого диску, яка може бути досить обмеженою і необхідною для встановлення різноманітних програм, застосувань, а також для правильного і швидкого функціонування пристрою. Таким чином виникає потреба в створенні бази даних для збереження фотографій та інших типів зображень, або так званих фотохостингів.

Фотохостинг – це веб-сайт який надає можливість зберігати фото та інші зображення в форматах .jpg, .jpeg, .bmp, .svg, .gif, .png тощо, прийняті для файлів з зображеннями. Початково були розроблені для фотографів, однак пізніше перейшли у використання звичайними користувачами.

Фотохостинг являє собою один із різновидів 'хмарних' сховищ даних. До переліку його переваг входять можливість швидко завантажити фото, зберегти

його, також є можливість поділитись ним в інтернеті, використовуючи унікальний Url, який присвоюється кожному зображенню при завантажені. Також використання сховищ дає можливість економити місце на жорсткому диску пристрою, що загалом і стало головним поштовхом створення 'хмарних' сховищ.

Нині фотохостингів досить багато і більшість із них надає безкоштовно певний об’єм пам’яті. Однак, більшість з них не надає можливості одразу сканувати або імпортувати дані з відеокамери в форматі RAW. Тому актуальною є задача створення застосування, яке б дозволило отримувати зображення напряму з джерела за допомогою TWAIN інтерфейсу.

Для реалізації даної задачі буде використано модуль scanner.js, якщо є сканер чи принтер, підключений до ПК. Окрім того є функція швидкого захоплення зображення, яке можна одразу завантажити на сервер, для цього в користувача має бути підключена веб-камера.

Завдяки даному застосуванню користувач зможе сканувати, створювати фото та зберігати фото в онлайн фотогалерею, а також переглядати їх.

1. ЗАДАЧА РОЗРОБКИ TWAIN ІНТЕРФЕЙСУ

Метою дипломної роботи є створення TWAIN інтерфейсу для веб-застосувань Caché.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- обрати засоби реалізації TWAIN інтерфейсу для веб-застосувань Caché;
- розробити архітектуру та структуру TWAIN інтерфейсу для веб-застосувань Caché;
- розробити TWAIN інтерфейс для веб-застосувань Caché;
- розробити архітектуру та структуру веб-застосування, яке використовує розроблений TWAIN інтерфейс для зберігання зображень в БД Caché;
- розробити веб-застосування, яке використовує розроблений TWAIN інтерфейс для зберігання зображень в БД Caché.

TWAIN інтерфейс має мати функції:

- підключення до модулю сканера;
- відображення відсканованого зображення;
- збереження зображення.

Веб-застосування, яке ілюструє роботу розробленого TWAIN інтерфейсу для зберігання зображень в БД Caché повинно мати функції:

- завантаження зображення;
- створення зображення шляхом доступу до веб-камери;
- відображення завантажених та створених зображень.

Потенційними користувачами TWAIN інтерфейсу для веб-застосувань Caché є розробники програмного забезпечення, яким необхідно мати можливість отримувати зображення з TWAIN пристроїв в браузері і завантажувати їх в БД.

Для розробки функціоналу TWAIN інтерфейсу була використана мова програмування JavaScript, HTML5 та CSS, JQuery. Для створення Rest API було використано серверну платформу Node.js для роботи з JavaScript та його фреймворк express.js, також були підключені такі модулі як scanner, ejs, cacheodbc, nodemon, multer, ba64.

2. АНАЛІЗ ІСНУЮЧИХ TWAIN ІНТЕРФЕЙСІВ ДЛЯ ВЕБ-ЗАСТОСУВАНЬ

Scanner.js дозволяє будь-якій веб-сторінці отримувати зображення зі сканерів за допомогою JavaScript у більшості браузерів, таких як Chrome, Firefox, IE, Microsoft Edge тощо. Підтримуються практично всі сканери, а scanner.js підтримує широкий спектр вихідних форматів: JPEG, багатосторінковий PDF, PNG і TIFF. У більшості випадків установка програмного забезпечення не потрібна.

На стороні клієнта підтримуються як 32-розрядні, так і 64-бітні браузери. Сервер може бути реалізований будь-якою мовою програмування, наприклад, C# ASP.NET, Java, Python, PHP, Ruby on Rail тощо.

Коли користувач викликає операцію сканування і застосування сканування не підключено, scanner.js відкриє діалогове вікно, що спонукає користувача завантажити та запустити програму сканування.

Підключити сканер можна двома способами:

- використовуючи посилання на офіційний сайт розробника, вказаного в документації;
- використовуючи bower, завантажити його на власний сервер і підключити з папки, наприклад:

  <script src="bower_components/scanner.js/dist/scanner.js"
  type="text/javascript"></script>

Щоб виконати сканування потрібно викликати функцію scanner.scan, яка приймає аргументи: callbackFunction, requestSpecification, useAspriseDialog, showScannerUI.

Щоб ініціювати сканування потрібно просто викликати scanner.scan, наприклад, при натисканні кнопки. Код, наведений на рисунку 2.1, ініціює сканування, що виводить зображення у форматі jpg.

```
function scanToJpg() {
   scanner.scan(displayImagesOnPage,
   {
      "output_settings" :
      [
         {
            "type" : "return-base64",
            "format" : "jpg"
         }
      ]
   }
   );
}
```

Рисунок 2.1 – Код для виконання сканування

3. ЗАСОБИ РЕАЛІЗАЦІЇ TWAIN ІНТЕРФЕЙСУ ДЛЯ ВЕБ-ЗАСТОСУВАНЬ CACHÉ

Важливим чинником під час розробки програмного продукту є вибір засобів програмної реалізації та технологій. Середовищем розробки програмної системи було обрано Visual Studio Code. Для створення графічного інтерфейсу системи використовувався веб-інтерфейс, з використанням JavaScript, CSS, HTML5 та JQuery.

Для побудови REST API було використано node.js та його фреймворк express.js. В якості бази даних було використано об'єктно-реляційну СУБД InterSystems Caché.

3.1 Середовище розробки Visual Studio Code

Visual Studio Code — це редактор вихідного коду, розроблений компанією Microsoft для Windows, Linux, MacOs.

Це легкий та швидкий редактор для кроссплатформеної розробки. Він містить необхідний початковий набір інструментів для роботи з Git, наприклад, термінал, який дає змогу швидко і зручно взаємодіяти з git, а також підсвітку синтаксису обраної мови, відладку, рефакторинг. Для веб проектів тут є одразу вбудована відладка коду, а для інших – можливість встановити розширення (плагіни).

Найчастіше використовуванні плагіни в веб-розробників це, наприклад, Live Server, який дозволяє розробнику одразу тестувати свій код без перезавантаження сторінки, ESLint. Також є плагіни для підтримки інших мов програмування чи фреймовркв. Загалом, майже всі плагіни були розробленні сторонніми розробниками та доступні через Visual Code Marketplace.

Також даний редактор дозволяє кардинально змінювати свій зовнішній вигляд, встановлювати теми, які можна завантажити чи обрати зі стандартних.

Цей редактор являється безкоштовним, розробляється як програмне забезпечення з відкритим вихідним кодом.

3.2 Мова програмування JavaScript

Мова JavaScript дозволяє клієнтському сценарію взаємодіяти з користувачем та створювати динамічні сторінки. Вона була обрана, оскільки являється дуже швидкою, оскільки запускається відразу в браузері клієнта. Поки він не вимагає сторонніх ресурсів, JavaScript не сповільнюється викликами на сервер.

Основні веб-переглядачі підтримують компіляцію JIT (just in time - щойно) для JavaScript. Це означає, що не потрібно компілювати код перед його запуском.

Наразі, кожен сучасний браузер має підтримку JavaScript. Тобто він інтерпретує код. Це безкоштовна технологія і вона не вимагає проходження будь-якої процедури встановлення чи налаштування. Описана мова широко популярна при розробці клієнтської частини. На відміну від PHP або інших скриптових мов, JavaScript можна вставити в будь-яку веб-сторінку. JavaScript може використовуватися в багатьох різних програмах через підтримку іншими мовами, такими як Pearl та PHP.

Також обрана мова являється платформо незалежною, тобто будь-який код JavaScript може бути виконаний на різних видах обладнання, для яких написана програма JavaScript.

Будучи мовою на основі подій, різні сегменти коду виконуються кожного разу, коли певна подія відбувається в JavaScript. Тобто сегмент коду виконується, коли користувач натискає кнопку або переміщує курсор миші на об’єкт. Це значно спрощує і прискорює процес розробки. Також ця мова підтримує можливості об’єктно-орієнтовного програмування.

Крім того можна розробити цілу програму JavaScript від frontend частини до backend, використовуючи лише JavaScript. Серверний JS – це розширена версія JavaScript, яка забезпечує зворотний доступ до баз даних, файлових систем та серверів. Javascript на сервері – це код JavaScript, який працює над локальними

ресурсами сервера. Він подібний до C # або Java, але синтаксис заснований на JavaScript. Хоч це і одна мова, однак можливості можуть бути різними. Наприклад, JavaScript на стороні клієнта не може отримати доступ до жорсткого диска клієнта, тоді як на стороні сервера програма може отримати доступ до жорсткого диска сервера без проблем.

Для того щоб обробити JavaScript використовується рушій. Схематично його принцип роботи показаний на рисунку 3.1

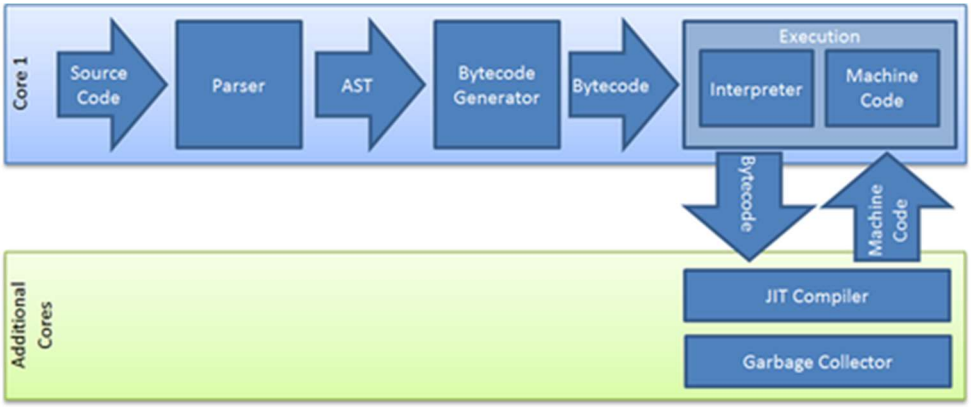


Рисунок 3.1 – Рушій для JavaScript

3.3 Платформа Node.js

Для створення серверної частини програми була обрана платформа Node.js, яка являє собою середовище виконання коду на JavaScript. Вона реалізована на основі рушія JavaScript Chrome V8, котрий перетворює виклики на мові JavaScript в машинний код, без необхідності його інтерпретувати.

Дана платформа була обрана з наступних причин. По-перше, вона дозволяє швидко перейти від frontend розробки, тобто розробки клієнт частини, до backend розробки, тобто розробки серверної частини, при цьому не вивчаючи нових технологій, оскільки використовується одна мова програмування – JavaScript.

По-друге, в Node.js є її пакетна екосистема npm, яка є найбільшою екосистемою бібліотек з відкритим вихідним кодом. npm постачає пакети, які можна встановити в проект і вони вирішують більшість відомих і частих проблем, тим самим роблячи розробку більш швидкою та ефективною.

Також Node.js дозволяє написати власні модулі, тобто незалежні частини програми, які можна використовувати в подальшому в інших проектах. Окрім цього, вона має набір вбудованих модулів, котрі можна використовувати без будь-якої ініціалізації.

Можливості Node.js можна легко розширити використовуючи різноманітні бібліотеки та фреймворки, яких на разі існує досить багато. Найбільш популярні серед них: Koa, Socket.io, Micro, Next.js, Meteor, Express.

Ще одними важливими перевагами Node.js є її асинхронність і подієво-орієнтований підхід, який на відміну від потокового програмування засновано на якихось подіях. Це можуть бути дії користувача, або, наприклад, подія надходження в програму мережевого пакету тощо. Схематично дана модель зображена на рисунку 3.2. В даній моделі один потік виконання обробляє події (наприклад, новий HTTP запит), що надходять в чергу, одна за одною, виконуючи відповідні обробники подій.

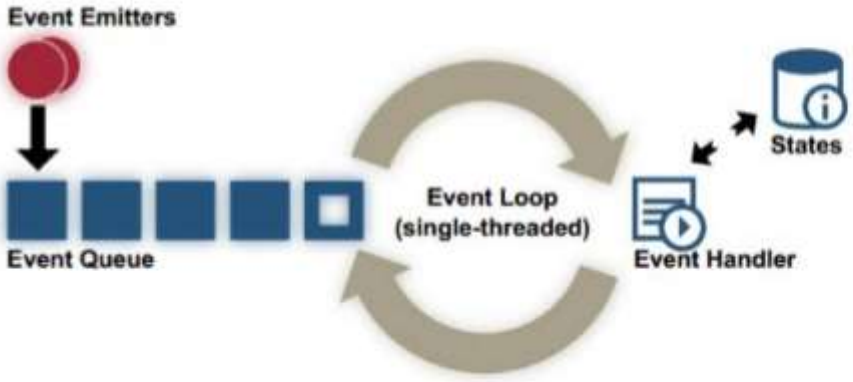


Рисунок 3.2 – Схематичне зображення подієво-орієнтованої архітектури

Архітектура середовища виконання складається з таких компонентів:

- рушій для мови JavaScript Chrome V8;
- бібліотеки для крос-платформенного введення-виведення та багатопотоковості libuv;
- бібліотеки для DNS (c-ares);
- бібліотеки для криптографії OpenSSL;

- бібліотеки для компресії даних zlib.

Схематично дану архітектуру зображено на рисунку 3.3

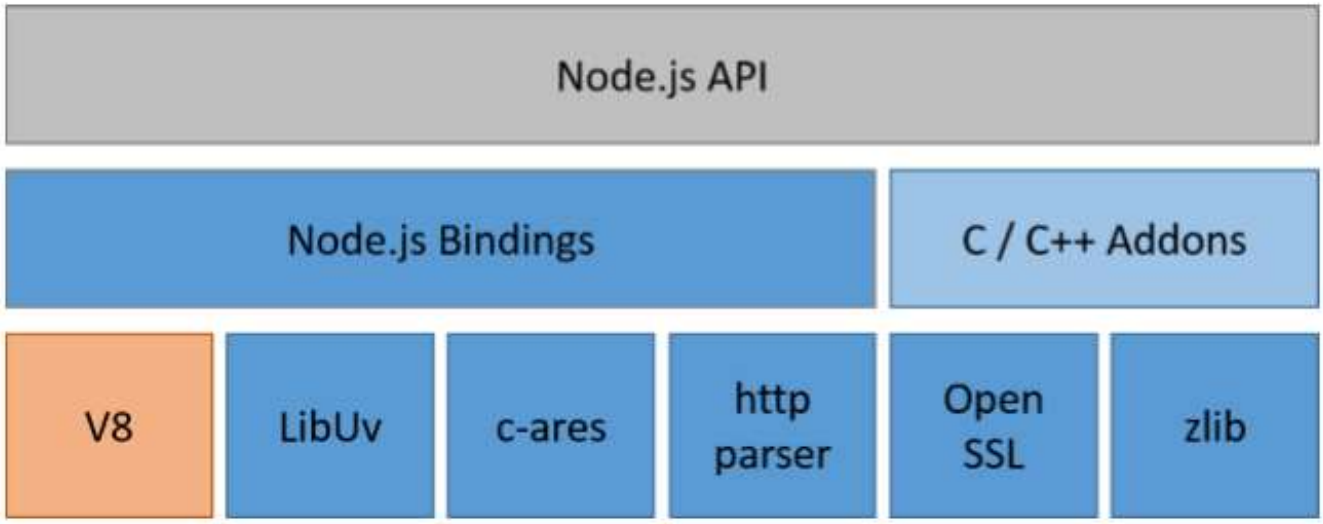


Рисунок 3.3 – Архітектура середовища виконання Node.js

3.4 Фреймворк express

В якості фреймворку було обрано express, так як він швидко освоюється і вважається одним із найпростіших фреймворків Node.js. До того ж він використовується досить давно, при цьому не втрачаючи своєї популярності.

Особливістю даного фреймворку можна визначати невеликий обсяг базового функціоналу. Таким чином, розробник отримує в своє розпорядження легкий та швидкий інструмент, який за необхідності можна розширити та розвинути за рахунок зовнішніх модулів підключених до проекту. При цьому немає ніяких обмежень, ні кількісних, ні функціональних.

Тож в результаті, даний фреймворк надає розробнику можливість бути вільним у виборі засобів вирішення тієї чи іншої проблеми. Хоча це і має певний мінус, оскільки передбачає більше роботи з боку програміста при виборі і організації модулів, але такий підхід фактично означає, що кожне застосування буде унікальним, оскільки немає універсальних рішень.

3.5 Мультимодельна СУБД Caché

Intersystems Caché — це сучасна система управління базами даних і середовище для швидкої розробки застосунків. Дана СУБД дозволяє істотно вдосконалити обробку і аналіз складних «великих даних», а також розробку мобільних і Web-застосунків. При цьому вимоги до апаратного забезпечення та адміністрування системи залишаються мінімальними.

Вбудованою мовою програмування для Caché є мова ObjectScript (COS), яка, в свою чергу, є надмножиною мови програмування MUMPS. Крім COS, Caché надає розробникам API для використання об'єктного і SQL-доступу до одних і тих самих даних.

Caché зберігає дані в багатовимірних масивах здатних містити ієрархічно структуровані дані. Схематично це зображено на рисунку 3.4. Ці ж "глобальні" структури застосовуються в мові програмування MUMPS; вони так само подібні до структур, що використовуються в MultiValue-системах.

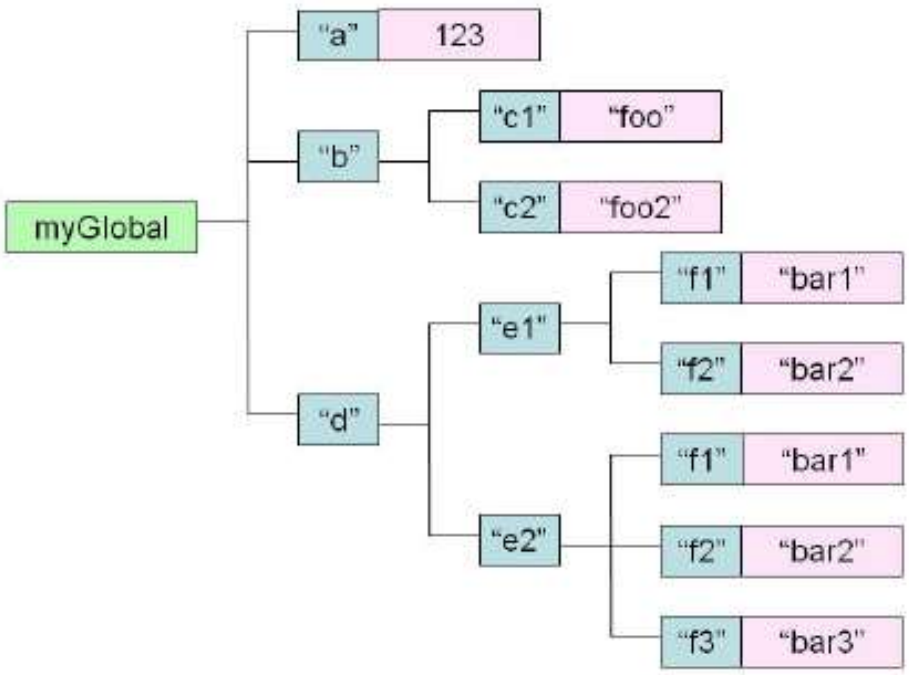


Рисунок 3.4 - Глобали СУБД Intersystems Caché

Зовнішні інтерфейси включають пряме зв’язування об'єктів C++, Java, EJB, ActiveX і .NET. Caché підтримує JDBC і ODBC для реляційного доступу. Також підтримуються XML і web-сервіси.

Дані в Caché зберігаються у вигляді розріджених масивів, що носять назву глобалів. Кількість індексів масиву може бути довільна. Це дозволяє описувати та зберігати структури даних довільного рівня складності. Індекси глобалів не мають типів, тобто можуть мати довільний тип даних.

Застосування розріджених масивів дозволяє оптимізувати використання об'єму жорсткого диска і скоротити час, необхідний для виконання операцій введення-виведення і вилучення даних.

В СУБД Caché реалізована розвинена технологія обробки транзакцій і вирішення конфліктів. Блокування даних проводиться на логічному рівні. Це дозволяє враховувати особливість багатьох транзакцій, які змінюють невеликий обсяг інформації. Крім цього, в Caché реалізовані атомарні операції додавання і видалення без блокування. Це знаходить застосування зокрема для лічильника ID об'єктів в базі даних.

Caché ObjectScript (COS) – це потужна об'єктно-орієнтована мова програмування, що вбудована в СУБД Caché. Синтаксис мови близький до синтаксису широко відомих мов програмування.

Об'єктна модель Caché відповідає об'єктній моделі стандарту ODMG (Object Data Management Group). Відповідно до ODMG кожен об'єкт Caché має єдиний визначений тип. Поведінка об'єкта визначається операціями (методами), а стан об'єкта — значеннями його властивостей. Властивості і операції визначають характеристики типу.

Система Caché підтримує наступні види зберігання об'єктів:

-автоматичне зберігання в багатовимірній БД;

-зберігання в користувацьких структурах даних;

-зберігання в таблицях зовнішніх реляційних баз даних, доступних через шлюз Caché SQL Gateway.

Відповідність між об’єктною та реляційною моделлю наведена на рисунку 3.5.

Об’єктне поняття	Реляційне поняття
Клас	Таблиця
Екземпляр	Рядок
Ідентифікатор (OID)	ID-стовпчик у вигляді первинного ключа
Властивість-константа	Стовпчик
Посилання на збережений об’єкт	Зовнішній ключ
Вбудований об’єкт	Окремі стовпчики
Колекція-список	Стовпчик з полем-списком
Колекція-масис	Підтаблиця
Потік даних	BLOB
Індекс	Індекс
Запис	Збережена процедура або представлення
Метод класу	Збережена процедура

Рисунок 3.5 – Відповідність між об'єктними та реляційними поняттями

ODBC – це стандарт API для доступу до БД, орієнтований на реляційні бази даних, стандартизований ISO/IEC (міжнародна організація, яка випускає стандарти). Забезпечує взаємодію між СУБД та користувацьким застосуванням. Загальну архітектуру ODBC наведено на рисунку 3.6.

Характеристики ODBC:

- інтерфейс рівня викликів;

- визначає стандартну граматику SQL;

- надає менеджер драйверів, щоб забезпечити роботу з одним або декількома драйверами та СУБД.

Архітектура ODBC має наступні компоненти.

1. Клієнт – виконує обробку даних та виклики функцій для обробки SQL запитів та отримання результатів.

2. Менеджер драйверів – завантажує та вивантажує драйвери за вимогою застосунків.

3. Драйвер – обробляє ODBC виклики та відсилає SQL запити до вказаного джерела даних.

4. Джерело даних – дані, до яких потребує доступ застосунок та відповідна операційна система, БД.

5. СУБД Caché реалізує стандарт ODBC 3.5, що дозволяє використовувати її в якості джерела даних ODBC.

Рисунок 3.6 – Архітектура ODBC

Для початку роботи з ODBC API необхідно встановити драйвер, який можна знайти на офіційному сайті InterSystems. Потім потрібно налаштувати DSN. Для цього в Панелі управління, обрати пункт Безпека -> Адміністрування, та обрати пункт ODBC Data Sources (32-bit), де:

- ім’я – задається користувачем для доступу до DSN через розроблений застосунок;
- опис – необов’язковий параметр, простір імен з яким користувач будете працювати;
- якщо не було вказане інше, ім’я користувача –_system, пароль – sys.
- порт на якому розташована СУБД 1972.

Необхідні налаштування вказано на рисунку 3.7.

Після налаштувань необхідно викликати в застосунку модуль та встановити з’єднання, SQLConnect, де першим параметром передається ім’я встановленого джерела даних, другий параметр – це функція обробник Error. На основі встановленого з’єднання можна надсилати запити, використовуючи метод .query, де перший параметр – це звичайний SQL запит, а другий параметр – функція обробки відповіді.

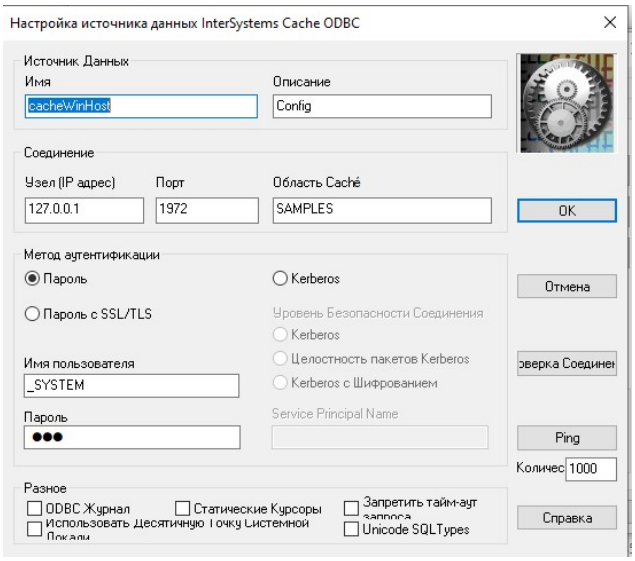


Рисунок 3.7 – Налаштування DSN

3.6 Бібліотека JQuery

Дану бібліотеку було обрано, так як вона фокусується на простій взаємодії JavaScript та HTML, дозволяючи легко отримати елементи DOM та маніпулювати ними. А також бібліотека дозволяє просто нарощувати функціонал завдяки додаванню різноманітних модулів. Обрана бібліотека є кроссбраузерною та її легко освоїти. Окрім цього вона має багато готових стандартних рішень, що значно економить час, якщо потрібно додати до проекту незначні деталі.

Бібліотека JQuery розвивається і підтримується завдяки добровольцям, на пожертвування. Таким чином, будь-хто може додати функціональність для маніпулювання веб-сторінками.

Архітектура даної бібліотеки зображена на рисунку 3.8.

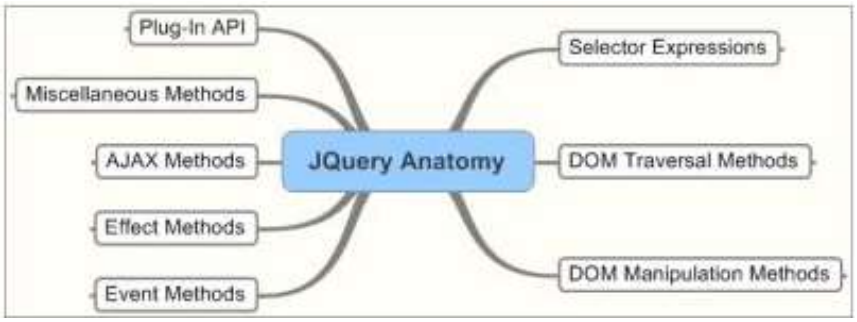
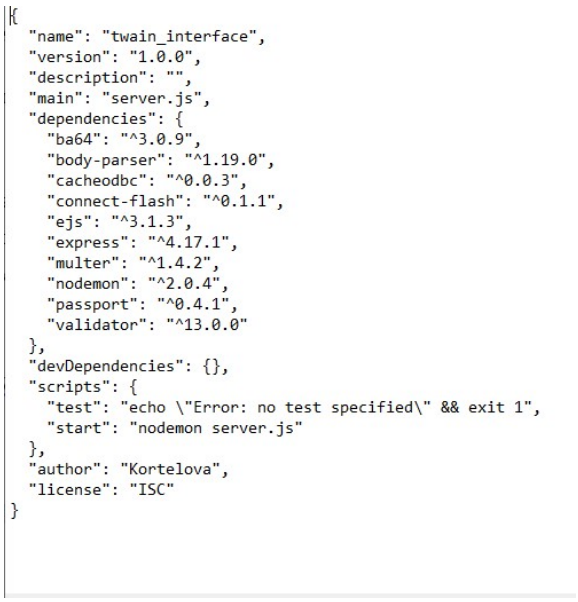


Рисунок 3.8 – Архітектура бібліотеки JQuery

3.7 Пакетний менеджер npm

Всі залежності описані в файлі package.json , який створюється при виконанні команди `npm init` всередині проекту. В файлі описано ім’я проекту, версія, модулі, які встановлені, короткий опис системи. Вони автоматично будуть додаватись до файлу package.json. В цьому і полягає основна меті даного файлу, щоб наступний розробник не розбирався в коді, шукаючи які модулі необхідно встановити, а зміг швидко і легко все побачити завдяки описаним залежностям. Щоб оновити пакети достатньо виконати одну команду ‘npm update’. Тобто використання опису залежностей в файлі спрощує роботу з залежностями і дозволяє легко ними керувати.

Файл має вигляд, приведений на рисунку 3.9.



```
{
  "name": "twain_interface",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "dependencies": {
    "ba64": "^3.0.9",
    "body-parser": "^1.19.0",
    "cacheodbc": "^0.0.3",
    "connect-flash": "^0.1.1",
    "ejs": "^3.1.3",
    "express": "^4.17.1",
    "multer": "^1.4.2",
    "nodemon": "^2.0.4",
    "passport": "^0.4.1",
    "validator": "^13.0.0"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon server.js"
  },
  "author": "Kortelova",
  "license": "ISC"
}
```

Рисунок 3.9 – Структура файлу package.json

При реалізації програмного забезпечення було використано наступні залежності.

Шаблонізатор EJS – це програмне забезпечення, яке дозволяє використовувати html шаблони для генерації html сторінок. Тобто можна легко додати невелику частину коду до сторінки html. Це спрощує читання коду, а також

дозволяє отримані дані з серверу одразу ж додавати до html сторінки. Наприклад, часто сторінку відповіді від серверу 404 генерують саме за допомогою шаблонів. Ejs був обраний оскільки він дуже простий та зрозумілий, до того ж, його синтаксис дуже наближений до чистого html. Зазвичай файли створені за допомогою шаблонізатора знаходяться в папці views. Вміст папки зображено на рисунку 3.10, оскільки node.js за замовчуванням виконує пошук шаблонів саме в цій папці

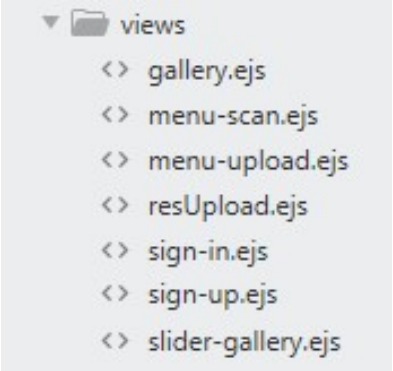


Рисунок 3.10 – Структура папки views

Nodemon – це інструмент, який допомагає розробляти застосування на основі node.js шляхом автоматичного перезапуску програми вузла, коли виявляються зміни файлів у каталозі. Тобто після кожної зміни на стороні сервера не потрібно вручну перезапускати сервер, це буде виконано автоматично.

ba64 – модуль npm для збереження закодованих зображень Base64, які є частиною URL-адрес даних у файловій системі. Використано для збереження зображення, яке було створено шляхом захвату з веб-камери та відображено на canvas.

Multer – це програмне забезпечення node.js для обробки форми multipart/form-data, яке використовується для завантаження файлів. Завдяки даному модулю завантажені файли за допомогою форми відправляються на сервер та файл зберігається в призначеній при конфігуруванні папці.

Validator – простий та зручний у використані модуль, який дозволяє перевіряти на вірність форми, тобто валідує їх, часто використовують, для перевірки чи дійсно введений емайл, а також для порівняння паролів.

3.8 Мова розмітки HTML та мова стилів CSS

HTML – Hyper Text Markup Language. Мова яка по суті будує ‘скелет’ для сайту. Загальна архітектура даної мови зображена на рисунку 3.11.

Майже всі сторінки у всесвітній павутинні описані за допомогою даної мови розміток. Спочатку мова HTML була задумана і створена як засіб структурування та форматування документів без їх прив'язки до засобів відтворення (відображення). В ідеалі, текст з розміткою HTML повинен був без стилістичних та структурних спотворень відтворюватися на обладнанні з різною технічною оснащеністю (кольоровий екран сучасного комп'ютера, монохромний екран органайзера, обмежений за розмірами екран мобільного телефону або пристрою і програми голосового відтворення текстів). Будь-який документ на мові HTML являє собою набір елементів, причому початок і кінець кожного елемента позначається спеціальними позначками – тегами. Елементи можуть бути порожніми, тобто не містити ніякого тексту та інших даних.

Нині використовується стандарт html5, який був випущений в 2014 році і надав набагато більше функціоналу, наприклад стали доступні такі теги як <video>, <audio> і <canvas>, які були використанні в розробці описуваного сервісу.

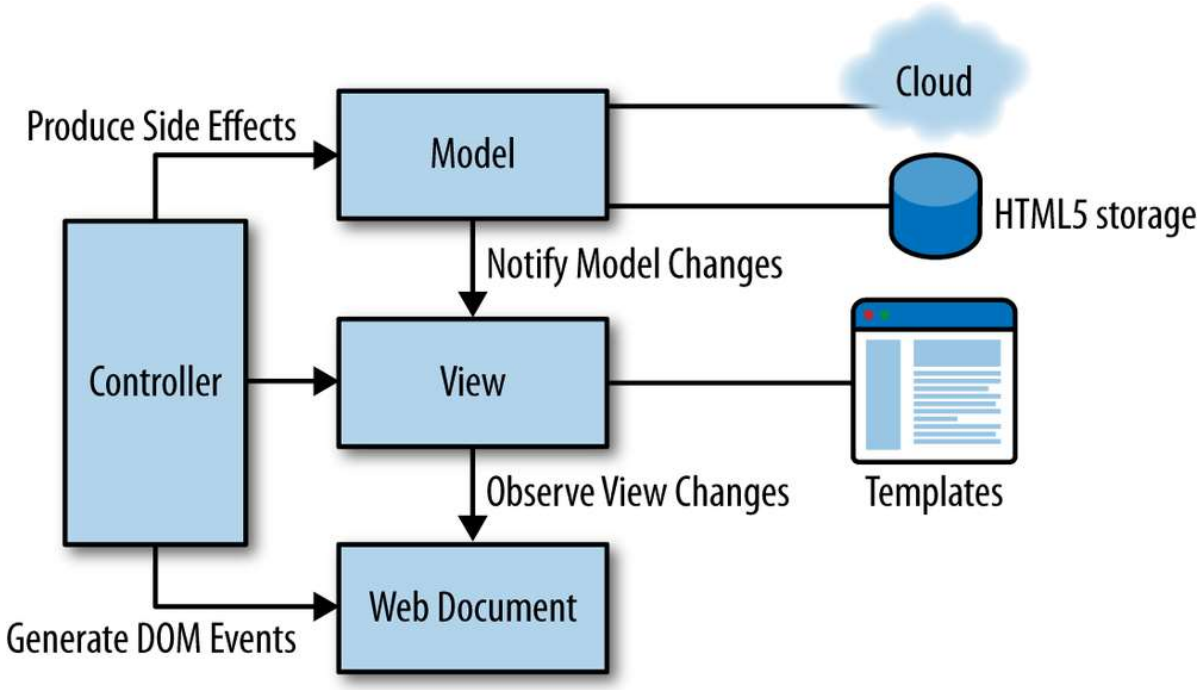


Рисунок 3.11 – Архітектура HTML

Cascading Style Sheets – каскадні таблиці стилів. Мова таблиць стилю, який дозволяє додавати стиль (наприклад, шрифти та кольори) до створеного документа (наприклад, документальний HTML та додаток XML). Зазвичай CSS-стилі використовуються для створення та редагування стильових елементів веб-сторінок та користувацьких інтерфейсів, написаних на мовах HTML та XHTML, а також можуть бути помічені у будь-якому відомі XML-документа, у тому числі XML, SVG та XUL. CSS спрощує створення веб-сторінок та обслуговування сайтів.

Каскадні таблиці стилів це технологія опису зовнішнього вигляду документа, оформленого мовою розмітки html. Каскадні таблиці стилів використовуються розробниками веб-сторінок для завдання кольорів, шрифтів, розташування і інших аспектів представлення веб-документа. Проста структура оголошення стиля наведено на рисунку 3.12.

Рисунок 3.12 – Структура оголошення стилю

4. ПРОГРАМНА РЕАЛІЗАЦІЯ TWAIN ІНТЕРФЕЙСУ

В даному розділі описуються програмні засоби, обрані для реалізації поставленої задачі, а також алгоритм роботи програмного продукту.

4.1 Алгоритм роботи TWAIN інтерфейсу

На рисунку 4.1 представлений алгоритм роботи TWAIN інтерфейсу.

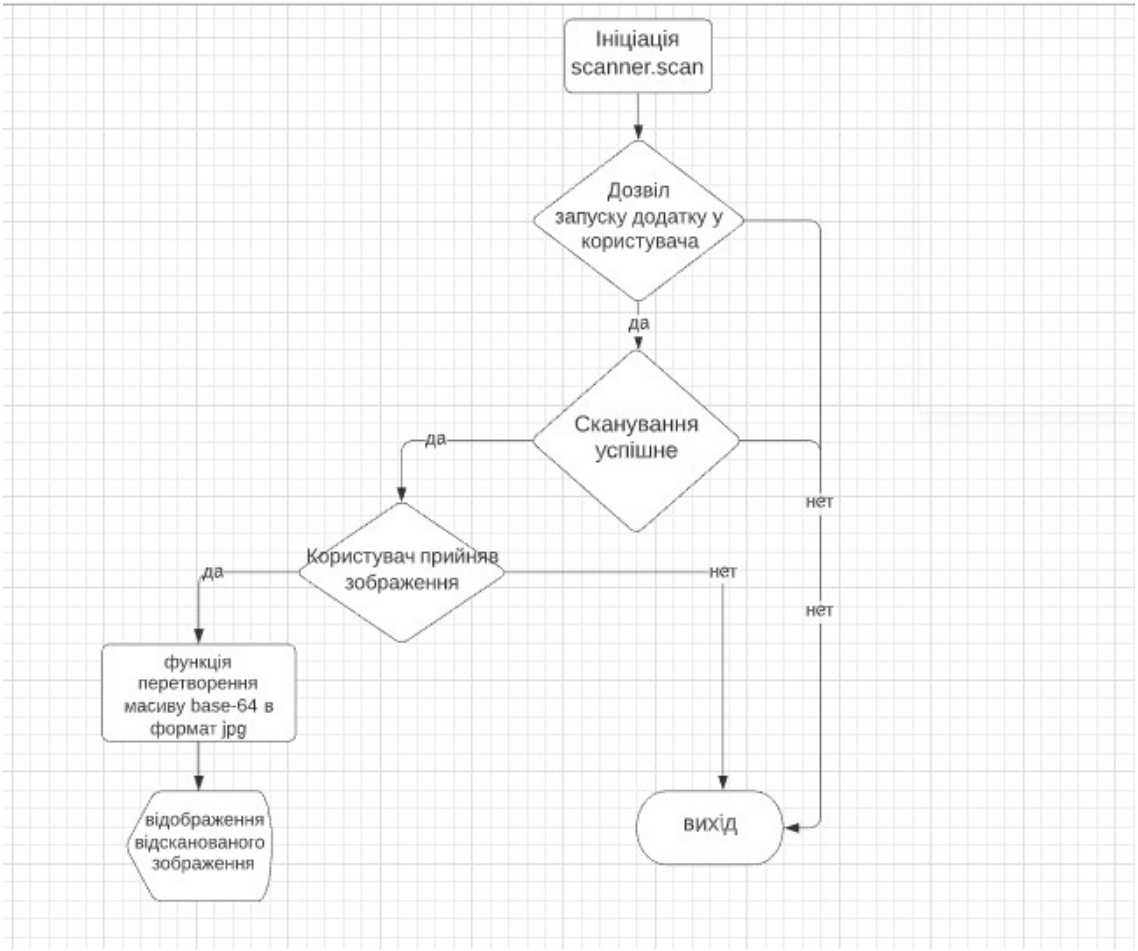


Рисунок 4.1 – Алгоритм роботи TWAIN інтерфейсу

Після запиту до сервера, щоб розпочати сканування, потрібно ініціалізувати сканер, за допомогою функції scanner.scan(), яка вбудована в модуль scanner.js.

Після чого необхідно отримати дозвіл від користувача, який дасть змогу відкрити застосування aspire scan. Без цього кроку неможливо запустити застосування в межах політики безпеки браузера. Якщо користувач відхиляє запит відбувається вихід з роботи програми.

Якщо запит підтверджено, відкриється десктопний застосунок, де можна встановити необхідні налаштування, обрати пристрій захоплення зображення та відсканувати потрібний документ. Якщо сканування завершилось успішно, то користувач побачить відскановане фото у вікні застосування.

Оглянувши прев'ю фото, можна завантажити відскановане зображення натиснувши відповідну кнопку. Після натискання цієї кнопки зображення конвертується в формат base-64, тобто кодування двійкових даних за допомогою латинських символів A-Z, a-z, 0-9, і передається на сервер, де за допомогою вбудованого методу модуля ba64 декодується в початковий формат jpeg.

Готове зображення відображається на сторінці користувача, де він може завантажити його до галереї, або ж знову обрати кнопку початку сканування і процес, описаний вище, запускається знову.

4.2 Архітектура та структура фотохостингу

Для розробки веб застосунку було обрано мову JavaScript. Зокрема на серверній стороні використано Node.js. Завдяки пакетному менеджерові npm, можна встановити модулі, для вирішення найросповсюдженіших питань.

Щоб модулі працювали їх необхідно встановити та викликати присвоївши в значення певної константи, наприклад const validator = require('validator')

В даному веб застосунку початковою є сторінка входу, де користувач вводить свій email та пароль. Для того, щоб перевіряти чи дійсно введено email, а не будь яка строка, використовується модуль validator, в нього є великий вибір методів, серед яких зокрема .isEmail(), де аргументом передається значення value поля input. Щоб на стороні сервера отримувати такі значення необхідно встановити модуль

body-parser, який надає доступ до полів форми через тіло запиту. Також модуль validator використано і на сторінці реєстрації, де необхідно перевіряти також пароль, а окрім того і те чи співпадають паролі введені користувачем, для цієї цілі використано метод .equels(), куди передаються два аргумента.

Після валідації користувача, необхідно зберегти введені дані в базу даних. Для доступу до бази даних було встановлено модуль cacheodbc, даний модуль надає odbc доступ, тобто можливість використовувати синтаксис sql Cache, до серверу баз даних Cache. Далі необхідно встановити сесію, для збереження сесії було обрано модуль client-sessions, який дозволяє зберігати сесію на стороні клієнта. При виході користувача відбувається розірвання сесії.

Авторизований користувач має доступ до завантаження фото на сервер. Для реалізації завантаження встановлено модуль multer, який дозволяє завантажувати фото з сервера в окрему папку. Також надається доступ до створення фото, за допомогою веб камери, отримане фото відображається на канвасі з заданими параметрами ширини та висоти. Щоб з канвас конвертувати в зображення необхідно використати метод модулю ba64 .writeImageSync(), однак попередньо потрібно обрати ім’я файла. Для цього взяти поточний час та дату та з’єднати їх в одну строку, проте користувач може змінити це ім’я при бажанні.

При відображенні галереї надається доступ до видалення фото, при натисненні кнопки видалення відправляється запит до бази даних із назвою зображення, та видаляється фото, після чого сторінка автоматично оновлюється. Якщо натиснути на будь-яке фото відкриється інший вигляд галереї, який було реалізовано завдяки функціям бібліотеки jquery.

Кожного разу, коли користувач обирає якийсь пункт меню, сервер йому повертає іншу сторінку. Для опису сторінки, яка буде рендеритись, було використано шаблонізатор ejs. Завдяки шаблонізатору влаштована галерея, цикл, який додає фото з бази даних написаний одразу на сторінці html, що значно зменшує код.

Також основний модуль, це фреймворк express, на ньому побудована вся серверна сторона.

Діаграма прецедентів зображена на рисунку 4.2, відображає дії які доступні для зареєстрованого користувача системи.

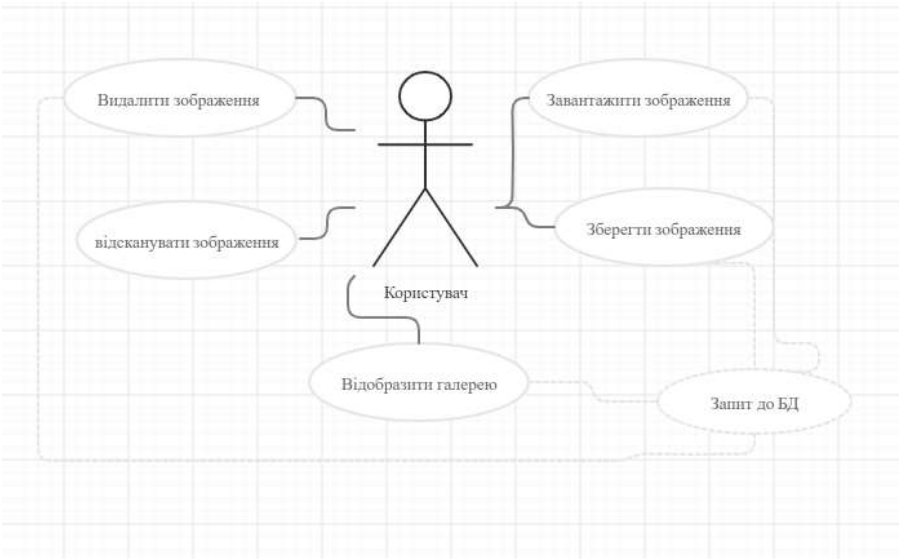


Рисунок 4.2 – Діаграма прецедентів

Початково вкладка галерея буде пуста тому, щоб поповнити галерею, користувач має на вибір 3 варіанти:

— завантаження фото зі свого девайсу;

— завантаження створеного фото за допомогою доступу програми до веб камери;

— завантаження відсканованого фото.

Сканування фото доступне в тому ж вікні, де надається доступ для створення живого фото.

Після того, як було завантажено певні зображення одним із описаних вище варіантів, користувачу надається можливість відобразити галерею. Де буде надано доступ до перегляду всіх фото, які буле завантажено, не важливо яким способом.

В вікні фотогалереї користувачу надається доступ до функції редагування фотогалереї, а саме видалення фото, а також надається доступ до зміни оригінального імені фото, яке було записано при завантаженні.

В описаному веб-застосуванні структура бази даних має вигляд, наведений на рисунку 4.3.

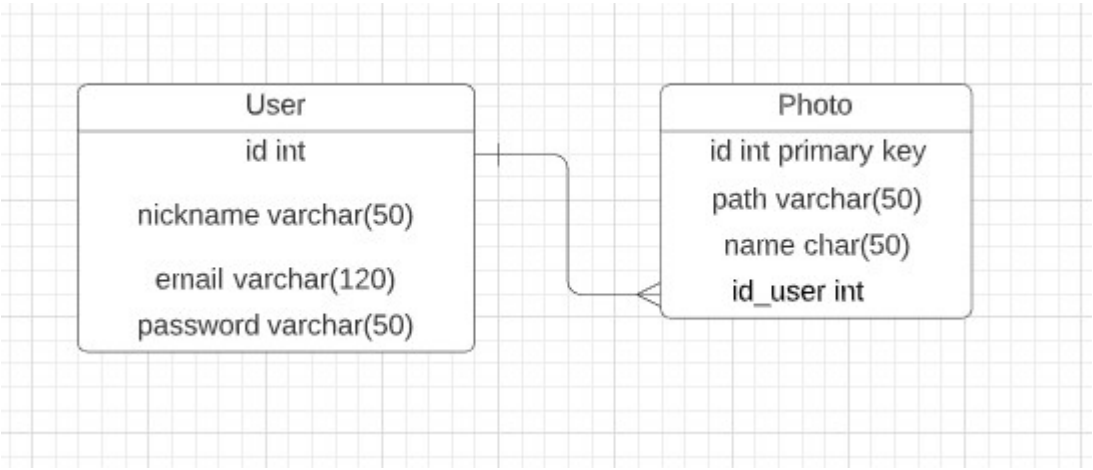


Рисунок 4.3 – Структура бази даних

Таблиця user містить в собі 4 поля для збереження інформації, про зареєстрованого користувача.

Поле email і поле password типу varchar, необхідні для входу користувача в систему. Кожного разу при спробі входу буде надіслано запит до бази даних, для вибору конкретних цих 2 полів і пошуку співпадінь.

Поле nickname, це унікальне ім’я користувача, необхідне для відображення в головному меню як привітання до конкретного користувача, а також відображається в розділі галерея.

Поле id являється primary key, тобто унікальним ключем. Також воно зв’язує дві таблиці між собою зв’язком типу 1:n, тобто один до багатьох, оскільки один користувач може мати необмежену кількість фото.

Таблиця photo містить чотири поля. Перше поле id, унікальний ключ, для якого обрано властивість auto increment, тобто автоматично встановлюється наступне значення поля більше на 1 від попереднього.

Поле id_user типу int потрібне для зв’язку фото з користувачем. Саме завдяки цьому полю в розділі галерея будуть виведені дані для користувача який ввійшов, а не абсолютно всі фото, які містять в базі даних.

В полі path збережено шлях до завантаженого файлу. При вставці картинки на веб-сторінку буде обрано всі значення path, де id_user співпадає з користувачем

для якого наразі триває сесія. Дані з цього поля будуть передані як масив і циклом for передані в атрибут src тегу img.

Останнє поле name зберігає назву фото. Оригінальну назву фото кристувач має можливість змінити, при цьому буде запит типу update до бази даних, та зміна цього поля.

5. МЕТОДИКА ВИКОРИСТАННЯ TWAIN ІНТЕРФЕЙСУ У ВЕБ-ЗАСТОСУВАННЯХ САСНÉ

В цьому розділі приведені системні вимоги для роботи з інтерфейсом та сценарії роботи користувача з ним.

5.1 Системні вимоги для роботи з TWAIN інтерфейсом

Для забезпечення коректної та безвідмовної роботи веб-сервісу персональний комп'ютер повинен мати процесор не гірше, ніж Intel ® Pentium ® / Celeron ® / Xeon™ або з тактовою частотою не менше 1,8 GHz або AMD 6 / Turion ™ / Athlon ™ / Duron ™ / Sempron ™ для користувачів процесорів від фірми AMD. Також на комп'ютері користувача має бути встановлений браузер.

5.2 Сценарії роботи користувача з фотохостингом

Перша сторінка при завантаженні localhost:8080 – це сторінка авторизації користувача для доступу до користування сервісом , її відображення подано на рисунку 5.1. На стороні серверу, завдяки модулю валідатора проводиться перевірка на правильність вводу, а потім в базі даних відбувається пошук заданого користувача, якщо такого не існує, сервер буде рендерити відповідь про помилку.

Якщо користувач не зареєстрований, зверху є посилання на сторінку реєстрації, яка має вигляд, як показано на рисунку 5.2.

Валідність форми перевіряється завдяки модулю validator. Використовуючи модуль body-parser з запиту пост виділяється тіло запиту, в якому записані необхідні об'єкти, такі як логін та пароль. Ці дані записуються в БД. Форма не буде відправлена поки не будуть заповненні всі поля, буде висвічуватись повідомлення про необхідність вказати дані, як зображено на рисунку 5.3. Авторизація можлива

завдяки збереженню сесії, тобто моменту коли користувач зайшов і використовує сайт, до моменту поки сесія не буде розірвана.

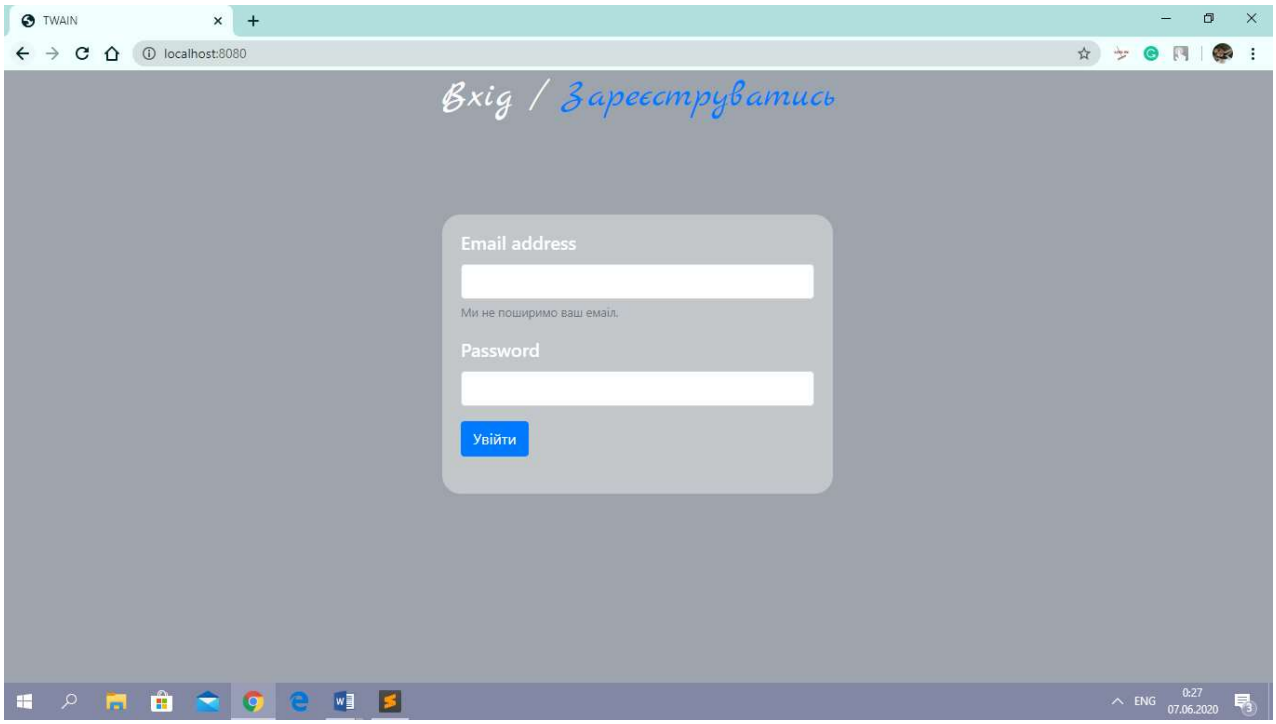


Рисунок 5.1 – Початкова сторінка сервісу

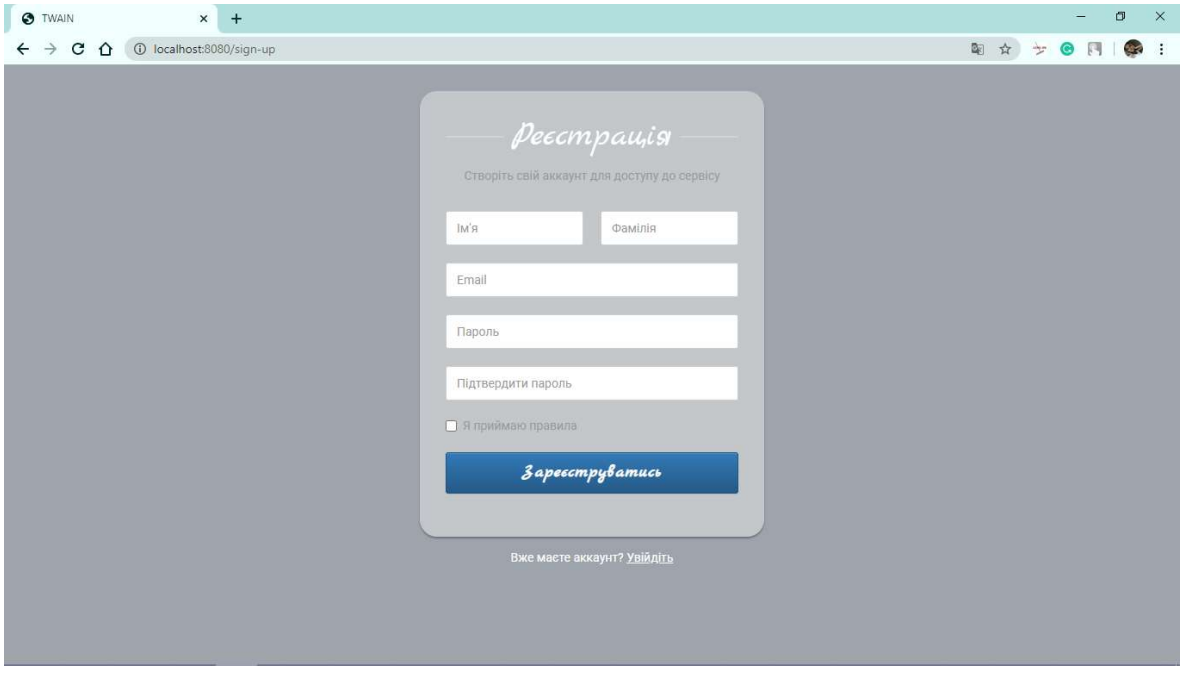


Рисунок 5.2 – Сторінка реєстрації

Головне меню веб застосунку доступне після авторизації користувача та має вигляд, поданий на рисунку 5.4 доступно 3 пункти меню, кожен із яких буде далі описано.

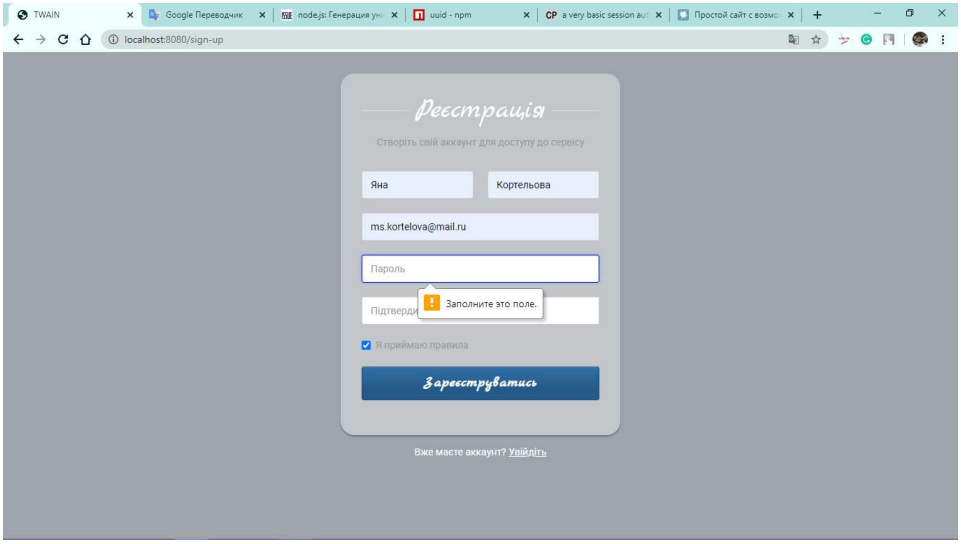


Рисунок 5.3 – Сторінка реєстрації

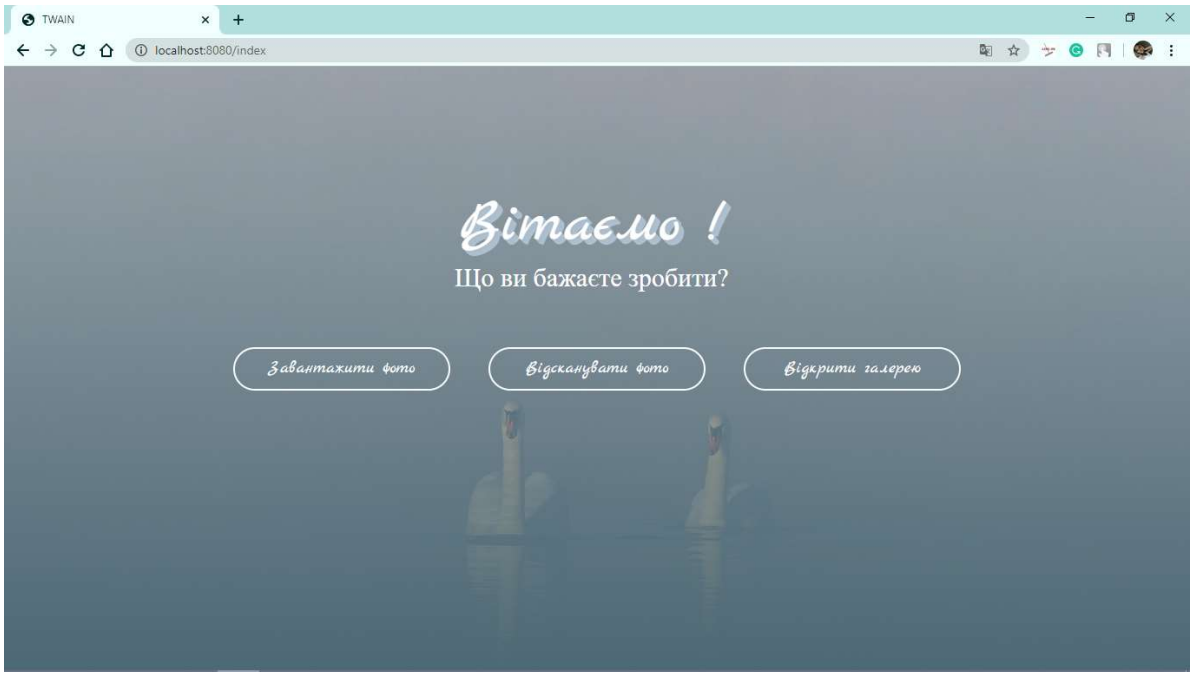


Рисунок 5.4 – Головне меню застосунку

Перший пункт – це завантаження фото на сервер. Тут доступна лише форма, яка відкриває папку для того, щоб користувач обрав, яке фото він хоче завантажити.

Перед тим як відправити фото на сервер, з’являється прев’ю цього фото, як зображено на рисунку 5.5. Також тут і в інших розділах сайту окрім головного меню доступна кнопка home, котра перенаправить користувача до головного меню.

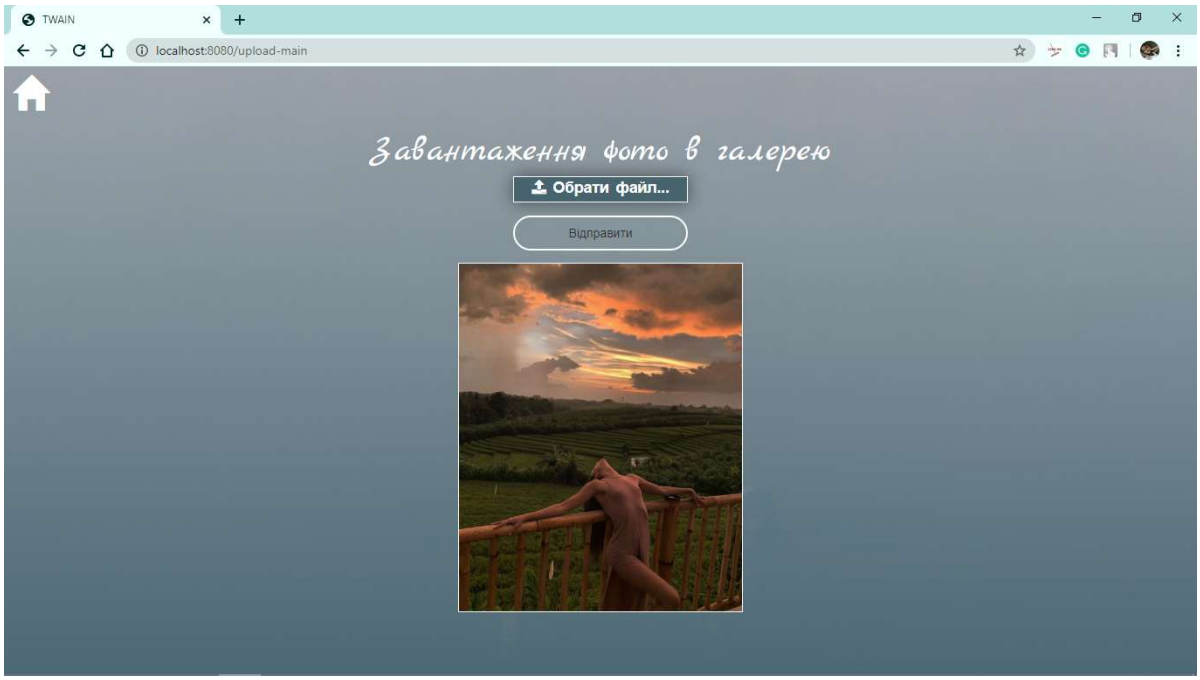


Рисунок 5.5 – Завантаження фото на сервер

Після натискання кнопки відправити буде сформовано відповідь від серверу. Якщо все пройшло без помилок, то користувач отримає відповідь про успішне завантаження (рисунок 5.6) і на вибір один з 2 пунктів меню.

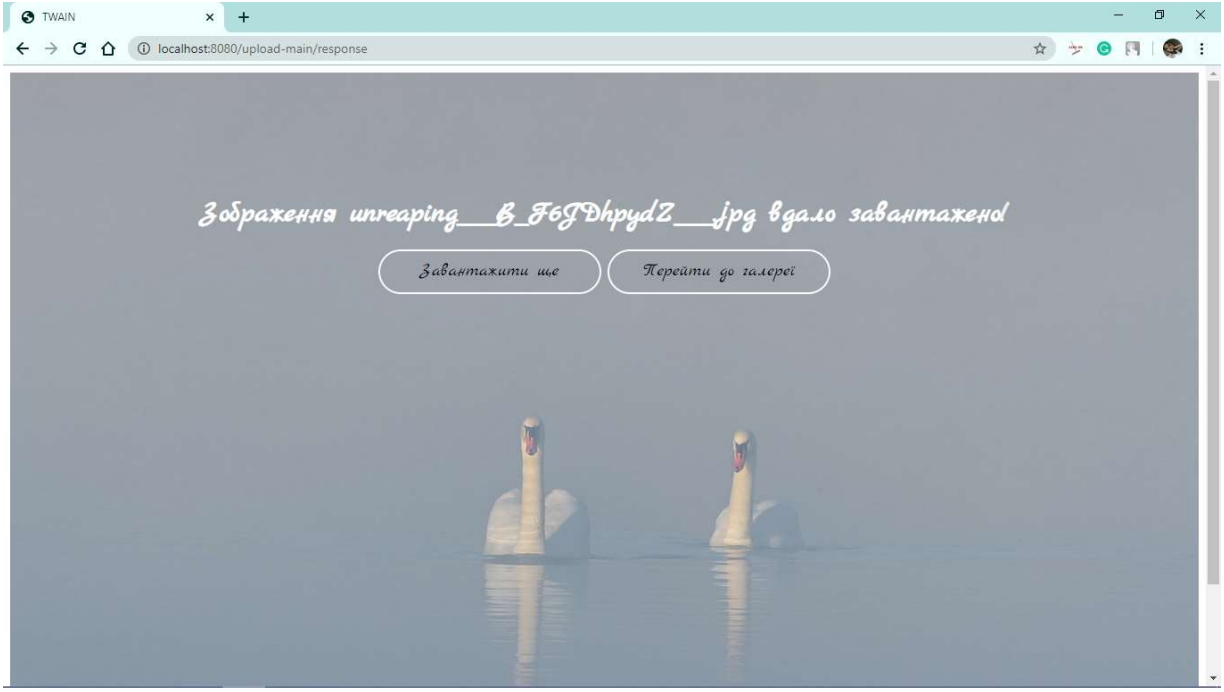


Рисунок 5.6 – Позитивна відповідь від серверу

Однак, якщо сталась помилка, або користувач натиснув, наприклад, відправити, однак при цьому не завантажив файл до форми, відповідь від серверу буде мати вигляд, поданий на рисунку 5.7.

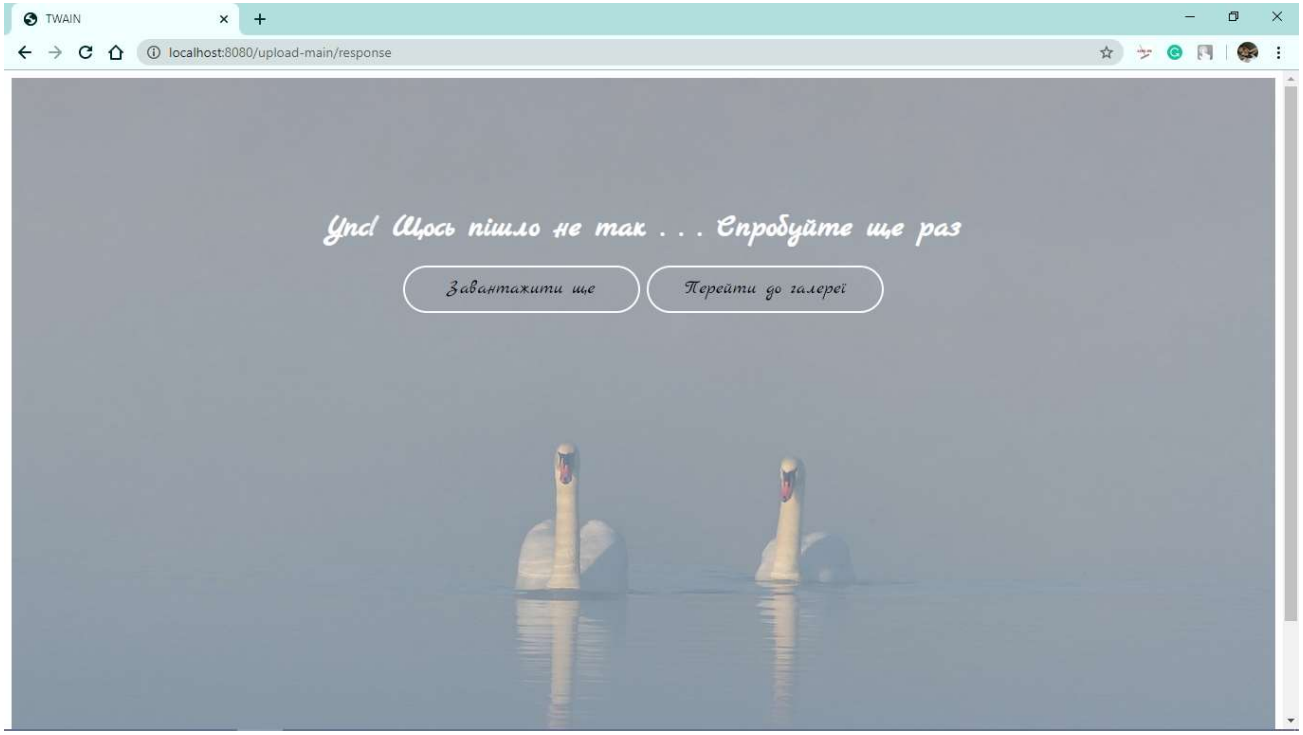


Рисунок 5.7 – Відповідь від серверу у разі помилки

Наступний пункт меню – це сканування фото , відображення якого зображено на рисунку 5.8. Тут доступний сканер, а також створення лайв фото, використовуючи веб-камеру свого пристрою. При першому натисканні кнопки буде надіслано запит на дозвіл використання веб-камери.

Створюючи лайв фото, воно переноситься на канвас і при натисканні кнопки завантажити до галереї, конвертується в формат base64, після чого передається на сервер, звідки перетворюється в звичайне фото і завантажується до бази даних. Приклад роботи даної функції показано на рисунку 5.9.

Пункт сканувати зображення надає доступ до модуля сканер, який був підключений. Спочатку буде відображено запит на підтвердження доступу до пристрою (рисунок 5.10).

Рисунок 5.8 – Меню сканування фото

Рисунок 5.9 – Створення лайв фото

Потім, якщо користувач прийняв запит, буде відкрите діалогове вікно, яке має вигляд приведений на рисунку 5.11, де можна провести всі необхідні налаштування та відсканувати фото, яке одразу ж буде завантажено до браузера.

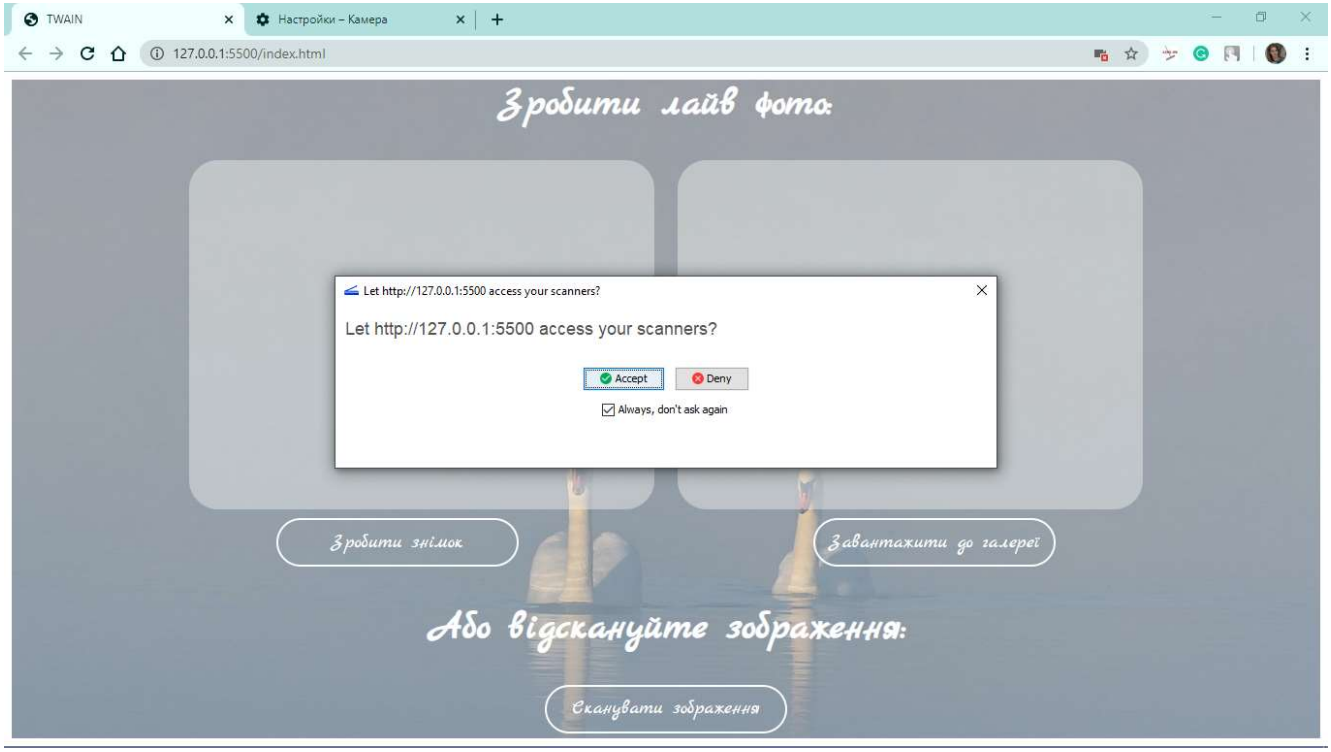


Рисунок 5.10 – Запит на підтвердження доступу

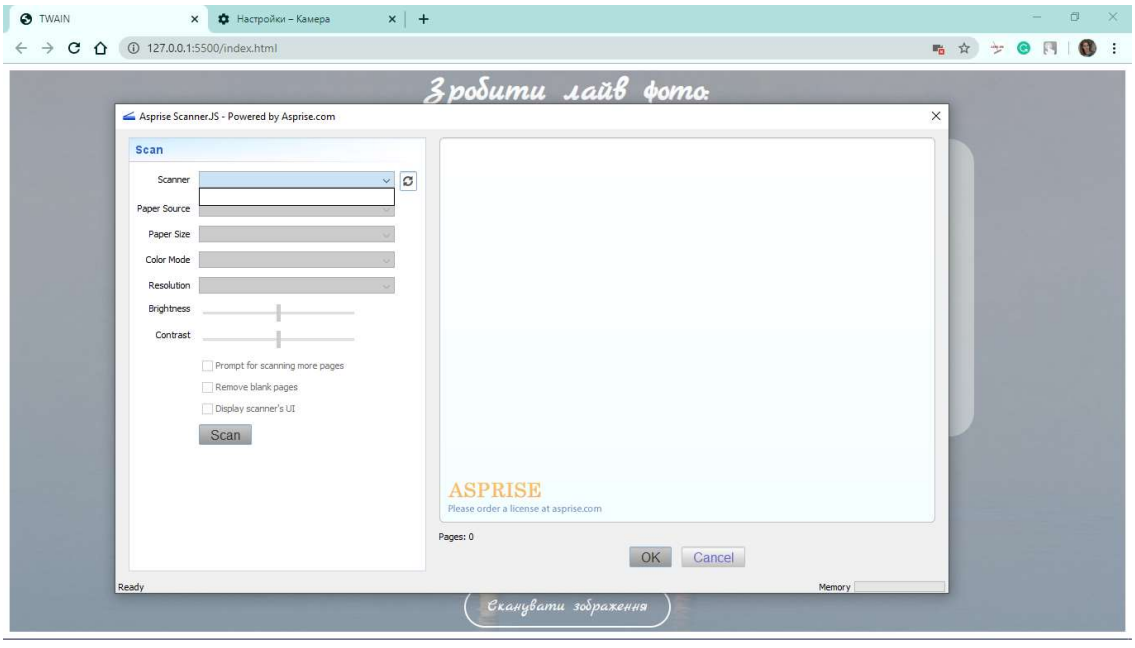


Рисунок 5.11 – Налаштування сканера

І останінй пункт головного меню – це фотогалерея, де зберігаються всі завантаженні фото. Відображення фотогалереї наведено на рисунку 5.12.

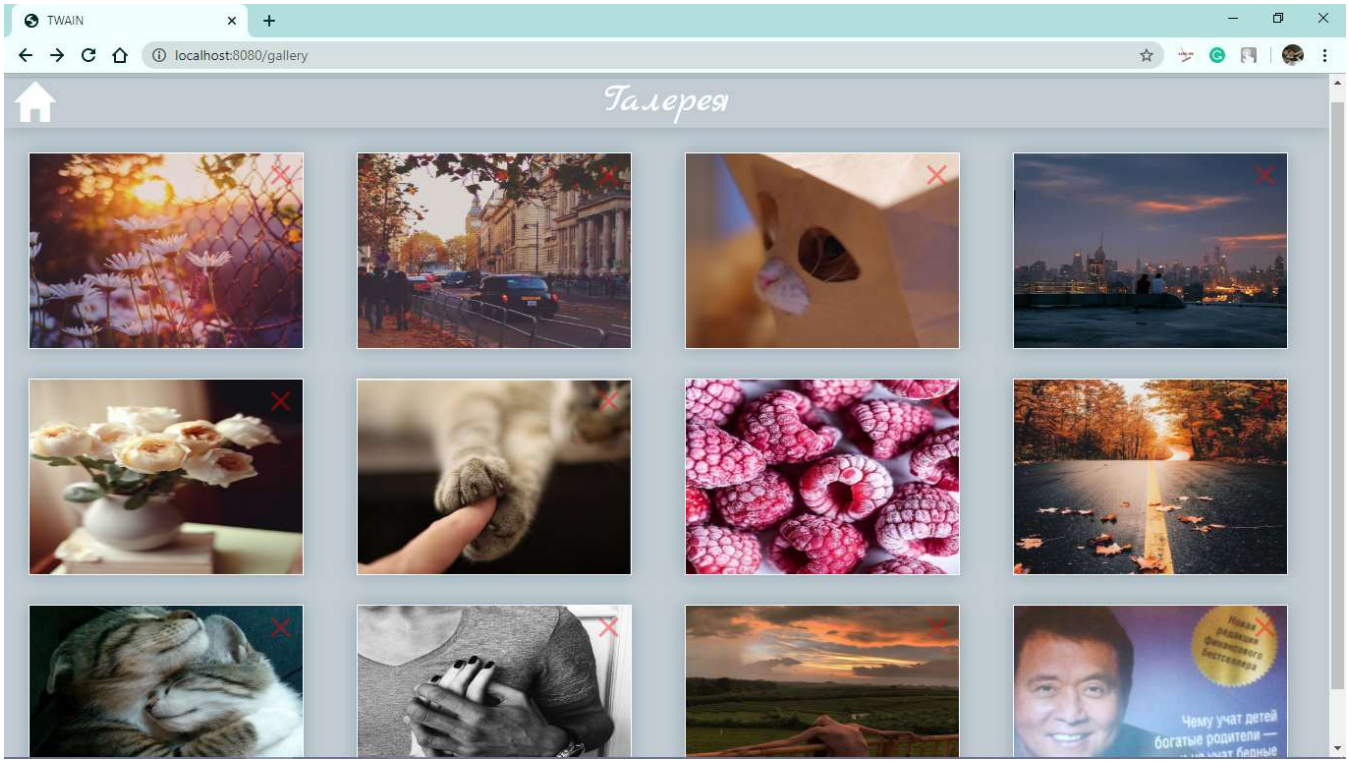


Рисунок 5.12 – Фотогалерея

На кожному фото є кнопка для видалення фото з галереї, при натисканні котрої буде створено запит, через url передано назву фото і запитом до бази знайдено та видалено фото з галереї.

Якщо натиснути на будь яке фото, буде відкрито слайдер, де можна переглянути фото в збільшеному розмірі (рисунок 5.13)

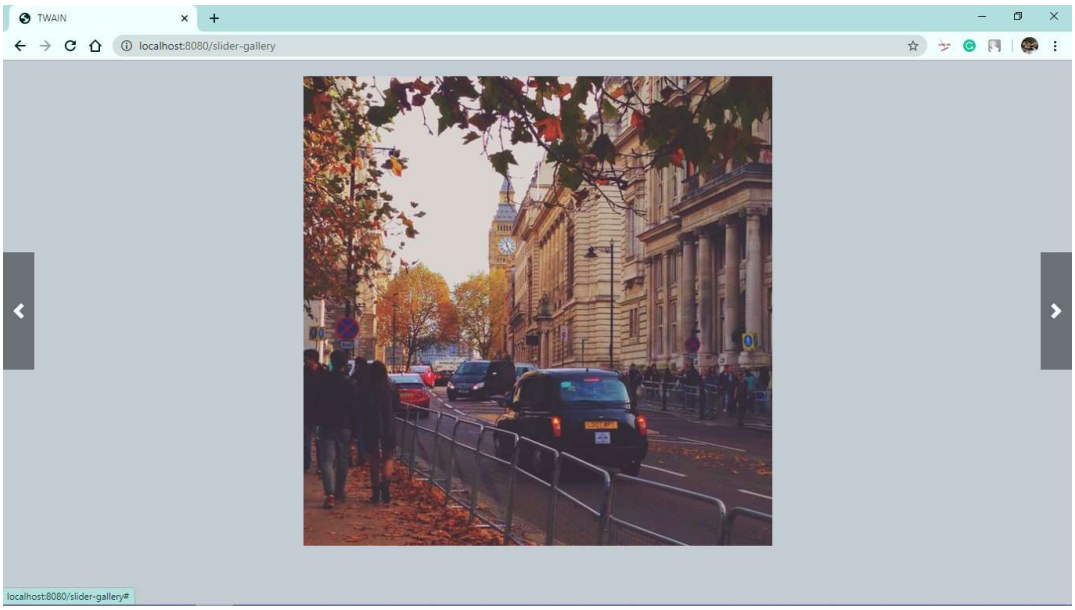


Рисунок 5.13 – Фотогалерея – слайдер

ВИСНОВОК

В ході виконання даної роботи було розроблено TWAIN інтерфейс для веб-застосувань Caché та веб-застосування для демонстрації роботи даного інтерфейсу.

Інтерфейс було написано використовуючи модуль scanner.js.

Сервіс фотохостинг було написано на мові програмування JavaScript, використовуючи додаткові інструменти розробки та базу даних.

Програму можна використовувати для сканування, завантаження, збереження та перегляду зображень в будь-який час.

В ході роботи було проведено огляд та зроблено аналіз засобів, що були використані для створення даного програмного забезпечення (середовища розробки Micrtosoft Visual Studio Code, Node.js і його фреймворк Express.js та засобів створення веб-інтерфейсів: HTML5, CSS, JavaScript).

Для роботи з даним програмним забезпеченням необхідний лише комп’ютер середньої потужності.

Користувач має змогу завантажувати свої фото, зберігати їх, використовуючи власну фотогалерею. Окрім цього надається функціонал сканування і завантаження фото одразу в вікно браузера. Також є можливість отримати фото на сайті.

Користувач може зберігати фото на особистий комп’ютер, або зберігати у базу даних та переглядати їх у будь-який час.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. About Node.js [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/about/
2. Aspire. JavaScript Scan in Browser [Електронний ресурс]. - Режим доступу: https://asprise.com/scan/scannerjs/docs/html/scannerjs-javascript-guide.html#the-scanning-process
3. Design Web Api [Електронний ресурс]. - Режим доступу: https://docs.microsoft.com/ru-ru/azure/architecture/best-practices/api-design
4. Express – Node.js web application framework – Режим доступу: https://expressjs.com/
5. Intersystems Cache [Електронний ресурс] – Режим доступу: https://cedocs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls
6. JavaScript MVC [Електронний ресурс] – Режим доступу: https://alistapart.com/article/javascript-mvc/
7. Sanders R.E. ODBC 3.5 Developers Guide. New York: McGraw-Hill, 1998.
8. Twain Specification [Електронний ресурс] – Режим доступу: https://www.twain.org/wp-content/uploads/2016/03/TWAIN-2.2-Spec.pdf
9. What is TWAIN [Електронний ресурс]. – Режим доступу: https://www.techopedia.com/definition/3631/twain 05.03.2020 р. – Заг. з екрану
10. Using Node.js with Caché 2018. [Електронний ресурс]. - Режим доступу: https://cedocs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=BXJS
11. Браун И. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript / Браун І. К: Питер, 2017. – 336ст.
12. Гайдаржи В. І. Об’єктно-реляційна СУБД Caché. Багатовимірний сервер даних і способи реалізації бізнес логіки засобами вбудованої мови Caché ObjectScript. Навч.посібн. / В. І. Гайдаржи, І. Ю. Михайлова. – К.: Освіта України, 2015. – 312 с.
13. Гамма Е. Приемы объектно-ориентированного проектирования. Паттерны проектирования. / Гамма Е., Хелм Р., Джонсон Р., Влиссидес Дж., - К: Питер, 2012. - 366 ст.
14. Еспозіто Д., Разработка современных веб-приложений: анализ предметных областей и технологий / Еспозіто Д., К: Williams, 2017 – 464ст.

15. Кантелон М. Node.js в действии / Янг А., Кантелон М., Мек Б., - К: Питер, 2018. – 432ст.
16. Михайлова І.Ю. Twain інтерфейс для веб-застосувань Cache/ І.Ю.Михайлова, Я.О. Кортельова // ІНФОРМАТИКА, МАТЕМАТИКА, АВТОМАТИКА ІМА::2020 : мат. міжнар. наук.-техн. конф., 20-24 квітня 2020 р. / М-во освіти і науки України, Сумський державний університет. – Суми : Сумський державний університет, 2020. – С. 151.
17. Маджуї М., Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте / Маджуї М., Уайлд Е., Мітра Р., - К: Питер, 2017 -232 ст.
18. Паван В. Шаблоны проектирования веб приложений / Паван В., -К: Ексмо, 2012 – 576ст.

ДОДАТОК 1

TWAIN інтерфейс для веб-застосувань Cache

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР6108_20Б

Аркушів 2

Київ – 2020

-2-

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61108_20Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61108_20Б 12-2	ba64.node	Робота з форматом base-64
	cacheodbc.node	ODBC підключення до бази даних
	ejs.node	Шаблонізатор
	express.node	Фреймворк
	multer.node	Робота з завантаженням на сервер
	nodemon.node	Тестування для серверу

ДОДАТОК 2

TWAIN інтерфейс для веб-застосувань Cache

Текст програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61108_20Б 12-2

Аркушів 8

Київ – 2020

-2-

```
//Server side
const express = require('express');
const app = express();
const fs = require('fs');
const multer = require('multer');
const cacheodbc = require('cacheodbc');
const connection = new cacheodbc.ODBCConnection();
const bodyParser = require("body-parser");
const ba64 = require("ba64");
connection.connect("DSN=cacheWinHost", err => {
  if (err) {
    throw new Error("Cannot connect to API");
  }
});
app.use('/public/images/', express.static('./uploads/'));

app.set('view engine', 'ejs');
app.use('/public', express.static('public'));

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false });
app.get('/', function(req, res) {
 res.writeHead(200, {'Content-Type': 'text/html; charset = utf-8'});
 let index = fs.createReadStream(__dirname + '/index.html', 'utf-8');
 index.pipe(res)
});
app.get('/index', function(req, res) {
 res.writeHead(200, {'Content-Type': 'text/html; charset = utf-8'});
 let index = fs.createReadStream(__dirname + '/index.html', 'utf-8');
```

-3-

```
index.pipe(res);
});

app.get('/upload-main',(req, res, next) => res.render('menu-upload'));

app.get('/menu-scan', (req, res, next) => res.render('menu-scan'));

app.get('/delete/:id', (req, res, next) => {
 let id = req.params["id"];

 connection.query(`delete from gallery where name = '${id}'`, (err, value) => {
  });
 res.send('Success');
 });

app.get('/sign-up', (req, res, next) => res.render('sign-up'));

app.post('/sign-up', (req, res, next) => {
 console.log(req.body);
 let pass = validator.equals(req.body.password, req.body.confirm_password);
 let email = validator.isEmail(req.body.email);
});

const storageConfig = multer.diskStorage({
  destination: (req, file, cb) =>{
    cb(null, "uploads");
  },
  filename: (req, file, cb) =>{
    cb(null, file.originalname.replace(/\s/g, ''));
```

```
 }
});

app.use(multer({storage:storageConfig}).single("filedata"));
app.post("/upload-main/response", function (req, res, next) {

 let filedata = req.file;

 if(!filedata)
   res.render('resUpload', {name: "});
 else{
  let name = filedata.originalname;

  res.render('resUpload', {name: name});
  connection.query(`insert into gallery values('${name}')`, (err, value) => {
    console.log(err);
  });
 }
});

app.post('/upload-canvas', (req, res, next) => {
 let data_url = ` ${req.body.photo}`;
 let time = new Date().toLocaleTimeString();
 var date = new Date().toLocaleDateString();
 let name = date+time;
 name = name.replace(/-/g, '');
 name = name.replace(/:/g, '');
```

-5-

```
ba64.writeImageSync(`uploads/${name}`, data_url, function(err){
  if (err) throw err;
 });
 connection.query(`insert into gallery values('${name}.jpeg')`, (err, value) => {
    console.log('cool');
 });
});

app.get('/gallery', (req, res, next) => {
 connection.query('select * from gallery', (err,value) => {
  let masName = [];
  for(let i = 0; i < value.length; i++){
   masName[i] = value[i].name;
  }
  res.render('gallery', {img: masName});
 });
});

app.get('/slider-gallery', (req, res, next) => {

 connection.query('select * from gallery', (err,value) => {
  let masName = [];
  for(let i = 0; i < value.length; i++){
   masName[i] = value[i].name;
  }
  res.render('slider-gallery', {img: masName});
 });
});
```

```
app.listen(8080);

//Scanner + Live video
document.getElementById("snap").addEventListener('click', () => {

 let video = document.getElementById('video');
 const mediaStream = new MediaStream();

 navigator.mediaDevices
  .getUserMedia({ video: true })
  .then((stream) => {
   this.video.srcObject = stream;
   return this.video.play();
  });


 let canvas = document.getElementById('canvas');
 let context = canvas.getContext('2d');

 document.getElementById("snap").addEventListener("click", function() {
   context.drawImage(video, 0, 0, 480, 360);
 });
});



//scanner module
let scanRequest = {
  "use_asprise_dialog": true, // Whether to use Asprise Scanning Dialog
```

```
 "show_scanner_ui": false, // Whether scanner UI should be shown
 "twain_cap_setting": {
  "ICAP_PIXELTYPE": "TWPT_RGB"
 },
 "output_settings": [{
  "type": "return-base64",
  "format": "jpg"
 }]
};

function scan() {
 scanner.scan(displayImagesOnPage, scanRequest);
}

/** Обробка результату сканування */
function displayImagesOnPage(successful, mesg, response) {
 if (!successful) { // On error
  console.error('Failed: ' + mesg);
  return;
 }
 if (successful && mesg != null && mesg.toLowerCase().indexOf('user cancel')
>= 0) { // User cancelled.
  console.info('User cancelled');
  return;
 }
 let scannedImages = scanner.getScannedImages(response, true, false); // returns
an array of ScannedImage
 for (let i = 0; (scannedImages instanceof Array) && i < scannedImages.length;
i++) {
```

-8-

```
        let scannedImage = scannedImages[i];
        let elementImg = scanner.createDomElementFromModel({
          'name': 'img',
          'attributes': {
            'class': 'scanned',
            'src': scannedImage.src
          }
        });
        (document.getElementById('images') ? document.getElementById('images') :
document.body).appendChild(elementImg);
      }
    }

    function saveImage() {
    var photo = canvas.toDataURL('image/jpeg');
    $.ajax({
     method: 'POST',
     url: '/upload-canvas',
     data: {
      photo: photo
     }
    });
    }

    document.getElementById("screen").addEventListener("click", saveImage);
```

ДОДАТОК 3

TWAIN інтерфейс для веб-застосувань Cache

Опис програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61108_20Б 13-1

Аркушів 8

Київ – 2020

АНОТАЦІЯ

Програмна система надає користувачеві можливість використовувати його в якості сховища для своїх зображень, та швидкого перегляду їх.

Розроблене веб-застосування можна використовувати в якості онлайн галереї, яка надає можливість, завантажувати та сканувати фото і зберігати їх до галереї, яка доступна в головному меню застосунку.

Програмний продукт розроблено за допомогою Node.js у парі з Express Framework, та мови програмування JavaScript.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Відповідно до назви дипломної роботи, розроблений веб-застосунок надає можливість підключатись до пристрою захвату зображення з вікна браузера, сканувати зображення та завантажувати їх до бази даних Cache.

Користувач для роботи з сканером має встановити Scan App від компанії Aspire.

Для використання сервісу потрібно лише встановити Scan App, не потрібно ніяких втручань, що потребують знань програмування.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений програмний засіб покликаний вирішити задачу відключення пристрою захвату зображення з вікна браузеру, а також захоплювати зображення використовуючи веб-камеру пристрою клієнта. Це було реалізовано за допомогою кількох функцій системи, а саме:

— підключення модулю для обробки і перетворення файлу формату base-64, в звичне зображення;

— підключення модулю scanner.js;

— підключення модулю для завантаження фото на сервер;

— завантаження даних про зображення в базу;

— визначення елементів які необхідно відобразити, використовуючи створену базу даних.

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Загальний принцип роботи застосування такий:

1) користувач обирає в головному меню пункт;
2) в залежності від обраного меню буде надана можливість сервер рендерить відповідне представлення;
3) якщо користувач обирає завантаження зображення, сервер оброблює запит, завантажує фото до папки, в базу даних записується шлях до папки;
4) коли користувач обирає відображення галереї, сервер виконує запит до бази даних, обираються необхідні дані, які віддаються в шаблон представлення, котре повертається користувачу.
5) При виборі пункту сканування фото, відкриється меню, де є можливість захоплення зображення, при натисканні кнопки зробити знімок, користувач вмикає свою камеру , яка робить захват зображення по натисненні тієї ж кнопки , відповідна картинка відображається на canvas, при натисненні зберегти до галереї , дана картинка конвертується в формат base-64, і на сервері даний формат конвертується в звичайне зображення, яке завантажується до папки, шлях до картинки також зберігається в базу даних;

-7-

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для користування веб-застосунком потрібно мати лише доступ до браузера, однак для використання можливості доступу до сканеру , необхідно встановити Scan App Aspire, який буде запропоновано встановити при першому натисканні кнопки сканування.

Користувач не обтяжений встановленням окремих програмних засобів та інструментів, які потребують спеціальних навичок.

-8-

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є:

— база даних шляху до папки з файлами;

— завантажене зображення;

— файл формату base-64.

Вихідними даними є:

— графічне відображення вибраних завантажених зображень.

ДОДАТОК 4

TWAIN інтерфейс для веб-застосувань Cache

Апробації

Аркушів 5

Київ – 2020


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАТИКА, АВТОМАТИКА, МАТЕМАТИКА
ІМА - 2020
МАТЕРІАЛИ
та програма
МІЖНАРОДНОЇ НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ
студентів та молодих вчених
(Суми, 20-24 квітня 2020 року)
Суми,
Сумський державний університет
2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАТИКА, МАТЕМАТИКА, АВТОМАТИКА

ІМА :: 2020

МАТЕРІАЛИ
та програма

МІЖНАРОДНОЇ
НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ
студентів та молодих вчених

(Суми, 20–24 квітня 2020 року)

Суми
Сумський державний університет
2020

44. WEB-орієнтована система обліку сервісного обслуговування транспортного засобу

    Автори: студ. **Ляшенко М.В.**,
    доц. Марченко А.В.

45. Web-орієнтована система підримки організації та проведення аукціону арт-галереї

    Автори: студ. **Маркова А.Л.**,
    доц. Марченко А.В.

46. Web-додаток для підтримки організації фотозйомки

    Автори: студ. **Охріменко В.О.**,
    доц. Марченко А.В.

47. Інформаційна система для аналізу збитків від техногенних або природніх катастроф

    Автори: студ. **Шишкін О.В.**,
    доц. Марченко А.В.

48. Використання технології Big Data у фінансовому секторі

    Автор – студ. Медяник Є.І.

49. Переведення чисел в текст українською мовою в Caché ObjectScript

    Автори: доц. Михайлова І.Ю.,
    студ. **Баранюк Б.О.**

50. TWAIN інтерфейс для веб-застосувань InterSystems Caché

    Автори: доц. Михайлова І.Ю.,
    студ. **Кортельова Я.О.**

51. Веб-застосування для завантаження та зберігання неструктурованих даних

TWAIN інтерфейс для веб-застосувань InterSystems Caché

Михайлова І.Ю., *доцент*; Кортельова Я.О., *студентка*
НТУУ «КПІ» ім. Ігоря Сікорського, м. Київ, Україна

На сьогоднішній день користувачі часто зустрічаються з необхідністю отримати зображення зі сканера або різних видів камер. Наприклад, сканери використовуються в магазинах для зчитування штрихкодів, охоронні компанії використовують дані камер зовнішнього спостереження для надання послуг із захисту майна тощо.

Для того, щоб була можливість взаємодії сканерів з програмним забезпеченням, був розроблений стандарт інтерфейсу TWAIN, який дозволяє встановлювати комунікацію між застосуванням та пристроями отримання цифрових зображень, такими як сканери, веб-камери, камери зовнішнього спостереження та цифрові камери. Він вирішує наступні питання: підтримує різні платформи та різні пристрої, надає можливість працювати з різними форматами даних.

Однак, прикладне програмне забезпечення має мати можливість обробляти інформацію, отриману зі сканера/камери з використанням TWAIN драйвера. Таким чином актуальним є питання створення інтерфейсу для збереження даних, отриманих від пристрою, в базі даних (БД).

Для вирішення цієї проблеми було запропоновано розробити клієнт-серверне веб-застосування на основі REST, яке дозволяє отримувати зображення з пристрою на клієнті та зберігати його в БД Caché. Система складається з таких частин: модуль для переведення фото / зображення зі сканера в цифровий формат, модуль для відправки та збереження файлу в БД Caché.

Для побудови web-серверу була обрана платформа Node.js – асинхронний сценарій виконання JavaScript. Основним модулем для розробки web-застосування використано фреймворк Express, для відображення інтерфейсу користувача – JavaScript, з використанням фреймворку React.js.

Завдяки даному програмному забезпеченню користувач може завантажити та зберегти свої фото чи картинки, отримані в результаті сканування документів, в базу даних, а також отримати їх для перегляду у цифровому форматі на екрані свого пристрою.